Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 176 (2020) 3536–3544

Procedia Computer Science

www.elsevier.com/locate/procedia

24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Automatic Anode Rod Inspection in Aluminum Smelters using Deep-Learning Techniques: A Case Study

Hamou Chehri[a], Abdellah Chehri[b,*], Laszlo Kiss[b], Alfred Zimmerman[c]

[a] *Bell-Canada, 671 Rue de la Gauchetière Ouest, Montréal, Québec, H3B 2M8, Canada.*
[b] *Department of Applied Sciences, University of Québec in Chicoutimi, Canada,*
[c] *Faculty of Informatics, Reutlingen University, Reutlingen, Germany.*

Abstract

Automatic fault detection using machine learning has become an exciting and promising area of research. This because it accurate and timely way to manage and classify with minimal human effort. In the computer vision community, deep-learning methods have become the most suitable approaches for this task. Anodes are large carbon blocks that are used to conduct electricity during the aluminum reduction process. The most basic function of anode rod inspection is to prevent a situation where the anode rod will not fit into the stub-holes of a new anode. It would be the case for a rod containing either severe toe-in, missing stubs, or a retained thimble on one or more stubs. In this work, to improve the accuracy of shape defect inspection for an anode rod, we use the Fast Region-based Convolutional Network method (Fast R-CNN), model. To train the detection model, we collect an image dataset composed of multi-class of anode rod defects with annotated labels. Our model is trained using a small number of samples, an essential requirement in the industry where the number of available defective samples is limited. It can simultaneously detect multi-class of defects of the anode rod in nearly real-time.

Keywords: Deep learning; Automatic Inspection; Anode; Industry 4.0

* Corresponding author. Tel.: +1- 418 545-5011; fax: +0-000-000-0000 .
E-mail address: achehri@uqac.ca

1877-0509
10.1016/j.procs.2020.09.033

1. Introduction

Aluminum has turned out to be the wonder metal of the industrialized world. No other single metal is so versatile in use and economics. Aluminum's growth rate is the highest amongst the significant basic metals today. However, aluminum production is highly energy-consuming. The major cost components of aluminum production are DC Power, Alumina, and Carbon anode. Aluminum is conventionally produced by the Hall-Heroult process in which alumina is dissolved in molten cryolite and electrolyzed with an intense direct current [1].

Artificial intelligence can very well be integrated into this industry evolution perspective. Indeed, the increase in sensors and, consequently, in data makes it possible to dream of a machine that prevents future breakdowns, even though they have the opportunity to perform automatic maintenance.

Artificial intelligence is, therefore, more and more present in all areas—even cases where human life is at stake, such as the medical.

Thirty years ago, there was a lot of excitement about using visual sensing in robotics. Artificial intelligence (AI) believed that this would be a technological revolution [2]. But it took about twenty more years until the software and hardware were developed enough to make a real difference.

Today, there is more and more talk of Industry 4.0. This process involves making a combination of robotics and digitization [3]. With the scanning technique, companies could monitor machines and equipment by installing multiple sensors [4]. With these sensors, it would be possible to improve the quality of the products. These sensors would also reduce or even eliminate downtime because, with the sensors, companies would be notified as soon as a machine needs maintenance. It would be possible to be notified in the event of breakdowns [5]-[6].

In industrial processes, one of the most critical tasks for ensuring the proper quality of the finished product is the inspection of the product's shapes. The traditional manual detection method is carried out manually, and workers are trained to identify shape defects. However, this method is very time consuming, inefficient, and can contribute to a severe limitation of the production, energy, and experience of the inspector [7]. To overcome the shortcomings of manual inspection, automatic shape defect detection based on machine vision comes into being [8].

With the rapid development of computer technology, machine vision has been widely applied in industrial production, especially for defect detection in industrial products. Over the last decade, a large number of surface defect detection algorithms have emerged. These algorithms can be roughly classified into three categories: Traditional methods based on image structure features, methods combining statistical elements with machine learning, and deep learning methods based on the Convolutional Neural Network (CNN) [9].

The traditional defect detection algorithm based upon image structure features mainly detects the surface defects by analyzing the texture, skeleton, edge, and spectrum of the image. The methods of combining statistical features with machine learning mainly extract statistical features from the defect surface. They then use machine learning algorithms to learn these features to realize surface defect detection. The CNN-based deep learning for surface defect detection is that CNN can simultaneously achieve the automatic extraction and recognition of elements in a network, and get rid of the trouble of manually extracting features [10]. The CNN-based deep learning makes defect detection more accurate, which provides a novel deep learning approach in the industrial field [11].

In recent years, machine learning has driven advances in many different fields [12]. We attribute this success to the invention of more sophisticated machine learning models [13], the availability of large datasets for tackling problems in these fields [14], and the development of software platforms that enable the easy use of large amounts of computational resources for training such models on these large datasets [15]-[16].

This paper attempts to train its convolutional neural network for shape defect inspection for an anode rod. First, a complete dataset of anode rod shape was established using SketchUp [17]. Six classes were created with five classes of defects forms. Next, the sample images were preprocessed through faster RCNN to realize the intelligent detection of shape defects. Finally, we will have a program that can identify and draw boxes around specific objects in pictures, videos, or in a webcam feed.

This paper is organized as follows. The related works on anode rod inspection is given in Section 2. In Section 3, we give an overview of the object detection techniques. The sampling and image preprocessing procedures are given in Section 4. Simulation results and pseudo-code were given in Section 5. Section 6 concludes the paper.

2. Related works

Tracking any objects in an actual video sequence is a very delicate task. Especially when these objects are non-rigid, the background of the scene is not fixed and in the case of several moving objects in the same scene. The task is more straightforward because the model and the constraints of variation of these objects are known a priori. Other geometric or static criteria can also be introduced in the identification of these objects.

The most basic is manual inspection with the use of gauges to assess the anode rod's geometry and stubs for subsequent fitment into the carbon anode [18]. While most smelters will have developed various physical inspection gauges for the required standards, the use of these gauges is a highly subjective operation. The operator's decision-making process on whether to reject a rod for repair will be heavily influenced by the present frequency of rod/stub damage, repair quotas, available production time, rod assembly/rodded anode stock levels, and rod repair cost pressures.

The components exist to assemble a fully automated rod/stub inspection, coding, and rejection system to eliminate the subjectivity associated with the operator's testing and decision-making during the rod inspection and rejection process. In addition to rejecting anode rods for traditional geometric defects, a fully automated system makes it possible to reject rods for repair based on the overall condition of the anode rod concerning all of the stubs working in parallel is not intuitive to operators (figure 1).

For example, one stub on an older assembly may be flagged for rejection, but the new stub will be shorter than the remaining stubs and result in a significant cast iron 'pancake' under the new stub. A fully automated system can calculate the net effect of all repair possibilities to any given anode rod and calculate the most cost-effective set of repairs to perform to that anode rod based on overall performance [18].

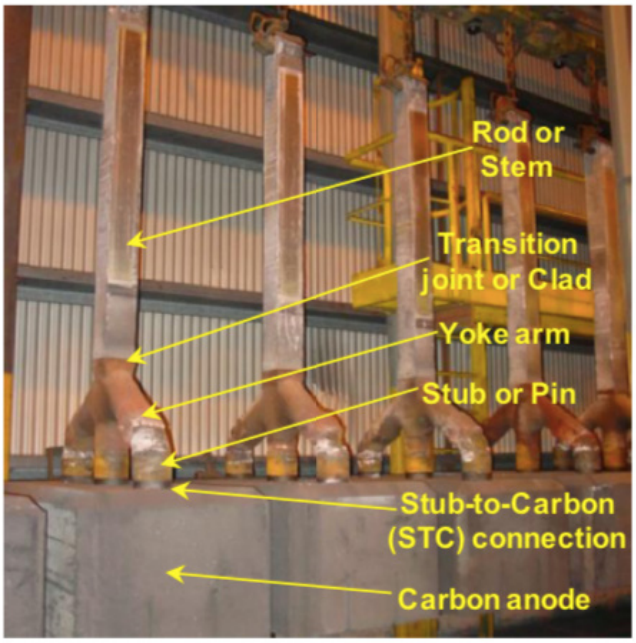

Fig. 1. Rodded anode assemblies [18].

3. Object detection techniques

The purpose of using object detection algorithms is to draw a bounding box around the object of interest to locate it within the image. You might not necessarily bring just one bounding box in an object detection case. There could be many bounding boxes representing different objects of interest within the image, and you would not know how many beforehand. Therefore, an algorithm like Faster R-CNN has been developed to find these occurrences and find them fast.

3.1. R-CNN Algorithm

R-CNN Algorithm has been developed to bypass the problem of selecting a considerable number of regions, Ross et al.[19] proposed a method where we use selective search to extract just 2000 regions from the image, and he called them region proposals. Therefore, instead of trying to classify a considerable number of regions, you can work with 2000 regions. These 2000 candidate region proposals are warped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output (figure 2).

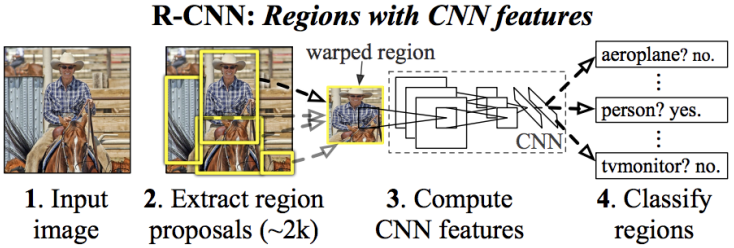

Fig. 2. R-CNN Algorithm [19].

3.2. Fast R-CNN

The same author of the previous paper (R-CNN) solved some of R-CNN's drawbacks to building a faster object detection algorithm, and it was called Fast R-CNN [19]. The approach is similar to the R-CNN algorithm (figure 3). Instead of feeding the region proposals to the CNN, we supply the input image to the CNN to generate a convolutional feature map. The reason "Fast R-CNN" is faster than R-CNN is because you don't have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is done only once per image, and a feature map is generated from it [19] – [20].

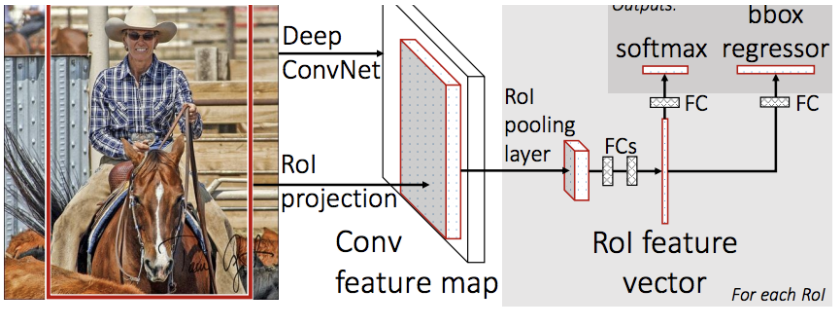

Fig. 3. Fast R-CNN [20].

3.3. Faster R-CNN

Both of the above algorithms (R-CNN & Fast R-CNN) use selective searches to determine the region proposals. Selective search is a slow and time-consuming process affecting the performance of the network. Therefore, Ren et al. [21] came up with an object detection algorithm that eliminates the selective search algorithm and lets the network learn the region proposals (figure 4).

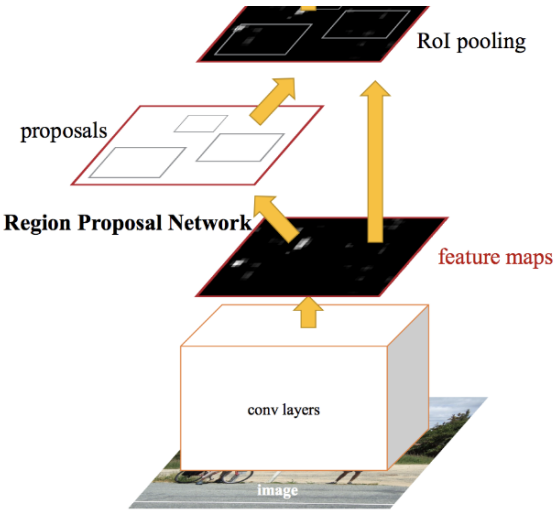

Fig. 4. Faster R-CNN [20].

Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

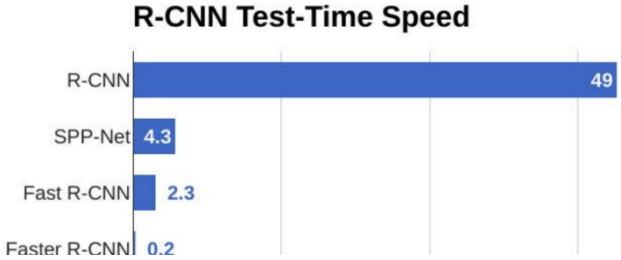

Fig. 5. Comparison of test-time speed of object detection algorithms [20].

From figure 5, you can see that Faster R-CNN is much faster than its predecessors. Therefore, it can even be used for real-time object detection.

Tensorflow detection model zoo provides a collection of detection models pre-trained. Some model has high speed with lower accuracy, other models such as Faster R-CNN have a lower speed but a higher efficiency.

4. Sampling and Image Preprocessing

4.1. Sampling

All implementation in this part has been done using an OpenCV environment. Pre-implemented functionality from the OpenCV library has been used to keep the program robust and will be stated when presented. This section will show a step-by-step solution of the image processing from source images to extracted features data of each leaf.

The data samples were collected by creating a 3D model of each anode rod using SketchUp. A total of 600 images were extracted (720×1280 pixels), 100 images for each class. The dataset contains the pictures of five classes of the defect and one class of regular anode rod. From the extracted images, 80% were randomly selected and allocated to the training set, and the remaining 20% were allocated into the test set.

4.2. Labeling

We used the LabelImg tool for labeling desired objects in every picture. Then draw a box around each object in each image. LabelImg saves a .xml file. This file will contain the label data for each image. These .xml files will be used to generate TFRecords, which are one of the inputs to the TensorFlow trainer.

4.3. Generate training Data

After labeling for all objects, TFRecords were generated, serving as input data to the TensorFlow training model. The image .xml data were used to create .csv files containing all the data for the train and test images. This creates a train_labels.csv and test_labels.csv file in the training folder.

4.4. Create label map and configure training

The last thing to do before training is to create a label map and edit the training configuration file. The label map tells the trainer what each object is by defining class names' mapping to class ID numbers. The label map ID numbers should be the same as what is defined in the generate_tfrecord.py file.

The fault detection for anode rod must be configured. It defines which model and what parameters will be used for training. There are several changes to make to the .config file, mainly changing the number of classes and examples, and adding the file paths to the training data.

4.5. Run and time the data training

Each step of training reports the loss. It will start high and get lower and lower as training progresses. For our training on the Faster-RCNN-Inception-V2 model, it started at about 1.6 and quickly dropped below 0.4. The model was left to train until the loss consistently drops below 0.05, which will take about 6265 steps (figure 6).

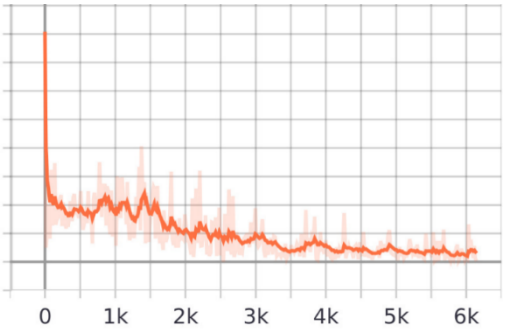

Fig. 6. One important graph is the Loss graph, which shows the overall loss of the classifier over time.

5. Simulation Result

The system architecture has proved to be a promising approach at this stage of development. All though there is still major parts of the system to be developed, its modularity makes it easy to develop and understand.

The implemented program works well and the training mode is fully functional. However, a classifier to be used in the spraying mode in the classification part of the system, is to be implemented. In this paper, the use of Python has been applied to examine the different classifiers. This is done in order to investigate the results and the performance before choosing a classifier and implementing it into the program. On the other hand, a framework for the program has been created and is easy to develop further.

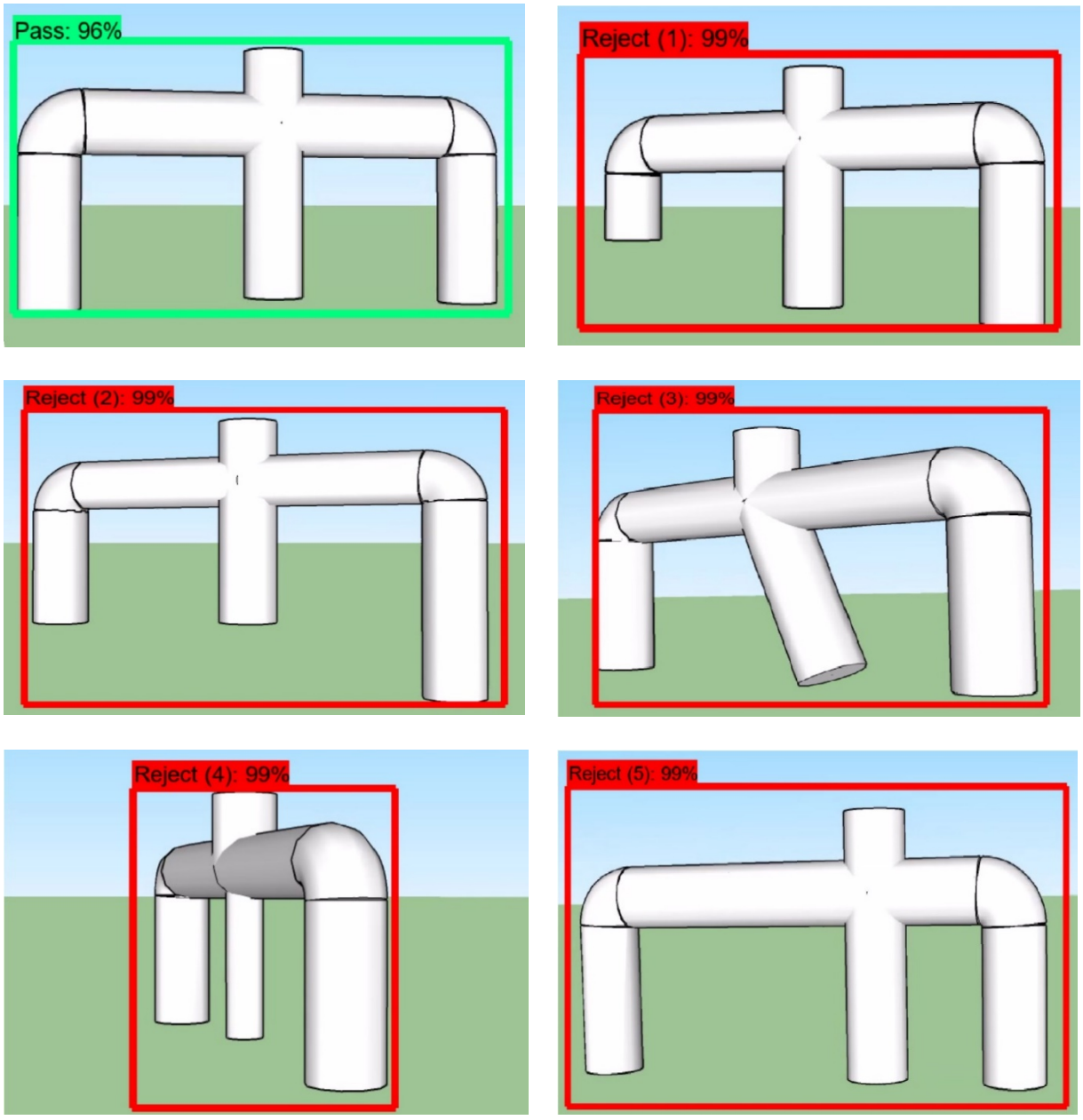

Fig. 7. Simulation results.

We created a variable in Anaconda uses Python 3.8 then we add these packages: pillow; lxml; Cython; contextlib2; opencv-python and cuda

Pseudo-code of the program:

```
# Import packages
import os
import cv2
import numpy as np
import tensorflow as tf
import sys
# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")

# Import utilites
from utils import label_map_util
from utils import visualization_utils as vis_util

# Name of the directory containing the object detection module we're using
MODEL_NAME = 'inference_graph'
IMAGE_NAME = 'testt (1).jpg'
# Grab path to current working directory
CWD_PATH = os.getcwd()
# Path to frozen detection graph.pb file, which contains the model that is used for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')
# Number of classes the object detector can identify
NUM_CLASSES = 6

# Load the label map.
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASSES,
use_display_name=True)
category_index = label_map_util.create_category_index(categories)
# Define input and output tensors (i.e. data) for the object detection classifier
# Input tensor is the image
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

# Output tensors are the detection boxes, scores, and classes
# Each box represents a part of the image where a particular object was detected
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
# Each score represents level of confidence for each of the objects.
# The score is shown on the result image, together with the class label.
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

# Draw the results of the detection (aka 'visulaize the results')
# All the results have been drawn on image. Now display the image.
cv2.imshow('Object detector', image)
# Press any key to close the image
cv2.waitKey(0)
# Clean up
cv2.destroyAllWindows()
```

6. Conclusion

The components exist to assemble a fully automated rod/stub inspection, coding, and rejection system to eliminate the subjectivity associated with the testing and decision making by the operator during the rod inspection and rejection process. In addition to rejecting anode rods for traditional geometric defects, a fully automated system makes it possible to reject rods for repair based on the overall condition of the anode rod concerning all of the stubs working in parallel, something that is not intuitive to operators.

For example, one stub on an older assembly may be flagged for rejection, but the new stub will be shorter than the remaining stubs and result in a significant cast iron 'pancake' under the new stub. A fully automated system can calculate the net effect of all repair possibilities to any given anode rod and calculate the most cost-effective set of repairs to perform to that anode rod based on overall performance.

References

[1] Amit Kumar Mishra, A rodded stepped stub anode assembly for an aluminum electrolytic cell, 2017.
[2] DeSouza, G.N. and Kak, A.C. (2002) Vision for Mobile Robot Navigation: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24, 237-267.
[3] Chehri A., Zimmermann A. (2020) Spectrum Management of Power Line Communications Networks for Industrial Applications. In: Zimmermann A., Howlett R., Jain L. (eds) Human Centred Intelligent Systems. Smart Innovation, Systems and Technologies, vol 189. Springer, Singapore
[4] A. Chehri and G. Jeon, "The industrial internet of things: examining how the IIoT will improve the predictive maintenance", Proc. Springer KES IIMSS 2019, pp. 517-527, June 17-19, 2019.
[5] A. Chehri and G. Jeon, "Routing Protocol in the Industrial Internet of Things for Smart Factory Monitoring" in Innovation in Medicine and Healthcare Systems and Multimedia. Smart Innovation Systems and Technologies, Singapore:Springer, vol. 145, 2019.
[6] Zoungrana WB., Chehri A., Zimmermann A. (2020) Automatic Classification of Rotating Machinery Defects Using Machine Learning (ML) Algorithms. In: Zimmermann A., Howlett R., Jain L. (eds) Human Centred Intelligent Systems. Smart Innovation, Systems and Technologies, vol 189. Springer, Singapore
[7] Tabernik, D., Šela, S., Skvarč, J. et al. Segmentation-based deep-learning approach for surface-defect detection. J Intell Manuf 31, 759–776 (2020).
[8] Patel, K. K., Kar, A., Jha, S. N., & Khan, M. A. (2012). Machine vision system: a tool for quality inspection of food and agricultural products. Journal of food science and technology, 49(2), 123–141.
[9] Zhou, F.; Liu, G.; Xu, F.; Deng, H. A Generic Automated Surface Defect Detection Based on a Bilinear Model. Appl. Sci. 2019, 9, 3159.
[10] Czimmermann, T., Ciuti, G., Milazzo, M., Chiurazzi, M., Roccella, S., Oddo, C. M., & Dario, P. (2020). Visual-Based Defect Detection and Classification Approaches for Industrial Applications-A SURVEY. Sensors (Basel, Switzerland), 20(5), 1459.
[11] Gopalakrishnan, K.; Khaitan, S.K.; Choudhary, A.; Agrawal, A. Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection. Constr. Build. Mater. 2017, 157, 322–330.
[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Ef- ficient estimation of word representations in vector space. In Proceedings of ICLR Workshops Track, 2013.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In Proceedings of NIPS, pages 1106–1114, 2012.
[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei- Fei. ImageNet Large Scale Visual Recognition Chal- lenge. International Journal of Computer Vision, 115(3):211–252, 2015.
[15] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ran- zato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng. Large scale distributed deep net- works. In Proceedings of NIPS, pages 1232–1240, 2012.
[16] Martín Abadi, et al. 2016. TensorFlow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation.
[17] https://www.sketchup.com
[18] Molenaar D., Sadler B.A. (2014) Anode Rodding Basics. In: Grandfield J. (eds) Light Metals 2014. Springer, Cham.
[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.
[20] Rohith Gandhi, R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms Understanding object detection algorithms, 2018.
[21] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks", Proc. 28th Int. Conf. Neural Inf. Process. Syst., pp. 91-99, 2015.